A CLOCK RATIO DATA SYNCHRONIZER

TECHNICAL FIELD OF THE INVENTION

The present invention is generally related to integrated circuits (ICs) and, more particularly, to a clock ratio data synchronizer for synchronizing data in one frequency domain to another frequency domain.

BACKGROUND OF THE INVENTION

Clock ratio data synchronizers are known and are used to synchronize data moving from one clock frequency domain to another clock frequency domain. For example, clock ratio data synchronizers are known that synchronize data moving from a higher clock frequency domain to a lower clock frequency domain, and vice versa. Currently clock ratio data synchronizers do not provide maximum tolerance between the skews of the clocks of the different clock frequency domains. Maximization of skew tolerance allows for more variation in the clock paths. For example, in cases where both clocks are being generated off-chip, variation between the off-chip clock paths will exist in addition to variation between the on-chip clock paths. As clock frequency increases, the skew tolerance margin (i.e., the maximum allowable difference between the skews) decreases.

With synchronizers used in non-state-of-the-art ICs, the skew tolerance margins are greater because the clocks run at lower frequencies than in state-of-the-art ICs. Consequently, skew tolerance margins were not a big concern. However, with the ever-

increasing demand to increase clock frequencies and data rates, current skew tolerance margins are insufficient using previous clock ratio data synchronizers.

Previous synchronizer designs utilize latches. Whatever is on the input of a latch is reflected on the output of the latch when the latch is in the "transparent mode". In the non-transparent mode of the latch, the output holds whatever value was on the output at the time that the signal that enabled the latch was removed. The transparent nature of the latch can potentially cause problems in the upstream logic in terms of the amount of time that the upstream logic has to make the input to the synchronizer valid. Therefore, it is possible that the upstream logic will not be able to meet setup time requirements. This also causes problems with the downstream logic because the downstream logic in the path of the synchronizer output will not always know when it must be ready to receive data, since the data could change at anytime during the time window over which the latch is transparent. These problems result in the aforementioned clock skew tolerance margin as well as setup time margin problems.

Accordingly, a need exists for a clock ratio data synchronizer that maximizes the skew tolerance margins between the clocks of the different domains, thereby allowing more variation to exist in the clock paths. A need also exists for such a synchronizer that will provide certainty with respect to setup time margins as well as potential increases in setup time margins. There is also a need for such a synchronizer that works well with static timing analyzers, and previous synchronizer designs presented many problems for static timing analysis.

SUMMARY OF THE INVENTION

In accordance with the present invention, a clock ratio data synchronizer is provided that utilizes a plurality of flip flops to synchronize data received by the synchronizer from first clock domain logic at a first clock frequency to a clock frequency of second clock domain logic. Each flip flop is capable of sampling data only on an edge of a clock. By utilizing flip flops in the synchronizer, as opposed to latches, data values are only allowed to change on clock edges. This, in turn, greatly improves the clock skew tolerance margin, and the setup time margins for the first clock domain logic and for the second clock domain logic.

These and other features and advantages will become apparent from the following description, drawings and claims.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram of a first example embodiment of an N-to-N-1 clock ratio data synchronizer in accordance with the present invention.

Fig. 2 is a set of waveforms drawn to facilitate the understanding of the example embodiment shown in Fig. 1.

Fig. 3 is a block diagram of a second example embodiment of an N-1-to-N clock ratio data synchronizer in accordance with the present invention.

Fig. 4 is a set of waveforms drawn to facilitate the understanding of the example embodiment shown in Fig. 3.

DETAILED DESCRIPTION OF THE INVENTION

Fig. 1 is a block diagram of a first example embodiment of an N-to-N-1 clock ratio data synchronizer 1 in accordance with the present invention. The data is input to the synchronizer 1 at input terminal 2 and is sampled by the higher clock domain, N. In this example, the ratio of N-to-N-1 is 5:4. Clock A corresponds to the higher clock domain and clock B corresponds to the lower clock domain. This means that for every 5 clock A cycles there will be 4 clock B cycles. In the higher frequency clock domain, it is known by the upstream logic that on a particular cycle, data cannot be transferred from the clock A domain to the clock B domain. This can be better understood from the timing diagram of Fig. 2.

Fig. 2 shows the relationship between clocks A and B 21 and 22, respectively, the Data In 23, which is data being clocked into the synchronizer 1 at the clock A rate, and Data Out 24, which is data being clocked out of the synchronizer 1 and the clock B rate. In the example shown in Fig. 2, the upstream logic knows that it cannot send the "B" data of the Data In 23 and therefore will either continue sending the A data that proceeds the B data for two clock A cycles or will send the A data for one clock A cycle and the C data for two clock cycles. This will occur whenever the rising edges of clocks A and B line up, which corresponds to edges 26A and 26B and edges 27A and 27B in Fig. 2. Thus, where clocks A and B are aligned at edges 26A and 26B, the D1 signal 25, which corresponds to the output of flip flop 5 in Fig. 1, will be held at the value of the A data for two clock A cycles. Similarly, where clocks A and B are aligned at edges 27A and 27B,

the D1 signal 25 corresponding to the output of flip flop 5 in Fig. 1 will be held at the value of the F data for two clock A cycles. Therefore, neither the B or G data received at the Data In terminal 2 will propagate through to the output of the flip flop 5, which corresponds to the D1 signal.

However the D0 signal is output from the flip flop 7 at the clock A rate because the flip flop 6 outputs a signal whenever the clock A signal is high and the flip flop 7 outputs a signal whenever the clock A signal is low due to the inverter 4. Since there are two latches 6 and 7 placed in series that receive the data from the clock A frequency domain, the output D0 26 will be delayed by one and one half clock cycles.

The manner in which the output of flip flop 5 is controlled to hold the output D1 25 of the flip flop 5 at a particular value for two clock A cycles is controlled by the enable signal, which is labeled 27 in Fig. 2. The enable signal 27 controls multiplexer 3. When the rising edges of clocks A and B are aligned, as is the case at edges 26A and B and 27A and B, the enable signal 27 is high just long enough to cause the A data from the Data In terminal (waveform 23 in Fig. 2) to be input to the flip flop 5. The enable signal 27 goes low before the next rising clock A edge and stays low until after that rising edge of clock A. This causes the output D1 25 of flip flop 5 to hold the input A data value for two clock A cycles, i.e., for the clock A cycles intended to gate in the A and B data. Thus, the waveform 25 corresponding to D1 shows no B value, but shows two A values followed by a C value.

The same occurs with D input value shown in the Data In waveform 23. The enable signal 27 stays high until just before the rising edge of clock A that precedes the rising edges 27A and B of clocks A and B that coincide. By keeping the enable signal 27

high during this period, the output D1 25 of flip flop 5 tracks the Data In input 23 of the synchronizer at the clock A rate. In the clock cycle before edges 27A and B, the enable signal 27 is low so that the D data value is held for two clock cycles. Just before the rising edges of clocks 27A and 27B coincide, the enable signal goes high for a long enough amount of time to gate the F value into flip flop 5 and then goes low until after the next rising edge of clock A, thereby causing the F data value to be held as the D1 signal 25 for two clock A cycles, i.e., for the F and G clock A cycles. Therefore, the D1 waveform 25 shows no G data value, but instead shows two sequential F data values.

The D1 and D0 data values are presented to the multiplexer 8. When the select signal 28 is low, the D0 data value is gated through to the output of the flip flop 9. When the select signal 28 is high, the D1 data value is gated through to the output of the flip flop 9. The output of flip flop 9 is shown in Fig. 2 and is represented by waveform 24. It can be seen that, due to the timing of the select and enable signals, the Data Out waveform 24 includes all data values input into the synchronizer 1 except for the B and G data values. It can also be seen by a comparison of the Data Out waveform 24 to the Data In waveform 23 that the Data Out waveform 24 corresponds to the clock B cycle whereas the Data In waveform corresponds to the clock A cycle. This is also apparent from the fact that the flip flop 9 is controlled by clock B.

The upstream logic has a signal going to it that tells it when the clock A and clock B edges are aligned, which allows the upstream logic to know that it can hold back the B data for one clock A cycle and that there is no need for the upstream logic to back up the B data.

In order to ensure that all of these things occur, the timing of the select signal 28 and of the enable signal 27 is important. The select signal 28 needs to be high prior to the rising edge of clock B that occurs when the rising edges of clocks A and B coincide. It then needs to go low before the rising edges of clocks A and B line up and remain low until after the next rising edge of clock B. It should be high at all other times. The low time and location remain fixed for different clock ratios, but the high time will vary. The enable signal 27 needs to be low prior to the rising edge of clock A that occurs one cycle before the rising edges of clocks A and B that are coincidental. It then needs to be high prior to the rising edge of clock A that occurs when the clock edges are coincidental. It then needs to be low prior to the rising edge of clock A that occurs one clock cycle after the clock edges coincide. It should be high at all other times. These timing relationships also ensure that the D1 data value 25 will not be changing as the Select signal 28 is changing.

Fig. 3 is a block diagram of the N-1-to-N portion 30 of the synchronizer of the synchronizer of the present invention. As in the example embodiment of Fig. 1, N=5 and N-1=4. Therefore, in this case, the data is being transferred from a lower clock frequency domain to a higher clock frequency domain. In this example, the C data is repeated for two clock A cycles and the clock A logic (i.e., the downstream logic) will take this into account based on knowing when the two clock cycles are in alignment (i.e., when the clock A and clock B edges are coincidental).

The circuit 30 shown in Fig. 3 will be described with reference to the timing diagram of Fig. 4 and with respect to the block diagram of Fig. 3. The circuit 30 comprises a first input flip flop 31, which receives the Data In and clock B signals 41 and

42, respectively, shown in the timing diagram of Fig. 4. On the rising edge of clock B, one of the data values shown in the Data In waveform 41 is sampled by the flip flop 31 and gated through to the output, Q, of the flip flop 31. This output is simultaneously made available to the multiplexers 32 and 33. The multiplexers 32 and 33 are controlled by the Enable 0 and Enable 1 signals 43 and 44, respectively.

When the Enable 0 signal 43 is high, the output of flip flop 31 is made available to flip flop 34. When the Clock A signal 45 goes low, the inverter 35 inverts the clock A signal, thereby causing the flip flop 34 to sample the data output from the multiplexer 32 and gates it to the output of flip flop 34. On the next rising edge of the clock A signal, the output of flip flop 34 is sampled by flip flop 36 and gated through flip flop 36 to its output D0, which corresponds to waveform 46 in Fig. 4. Therefore, the D0 signal 46 corresponds to the Data In value delayed by two clock A cycles.

When the Enable 0 signal is low, the output of the flip flop 34 is held at its current value. When the clock A signal 45 goes high, the value being held at the output of flip flop 34 is again sampled by flip flop 36 and gated to the output of flip flop 36, which corresponds to the D0 signal 46. Therefore, while the Enable 0 signal 43 is low, the value of D0 will remain the same.

As shown in Fig. 4, the Enable 0 signal 43 is high just before and just after the edges of clocks A and B coincide. Specifically, the Enable 0 signal 43 goes high prior to the falling edge of clock A that occurs just before when the rising edges of clocks A and B are coincident. It goes low prior to the second falling edge of clock A that occurs after the coincident rising edges of clocks A and B. The high time is fixed at two clock A

cycles, but the low time will vary with the ratio of the clock frequencies. The coinciding clock edges are represented in Fig. 4 by numerals 51A and B and 52A and B.

The timing of the Enable 0 signal 43 with respect to the coincidental rising edges 51A, 51B, 52A and 52B of clocks A and B causes the D0 signal 46 to maintain the A data value of Data In 41 for four clock A cycles (i.e., for the A data value cycle and for the B and C data value cycles of the Data In 41). Identically, this timing causes the D0 signal 46 to maintain the E data for the E data value cycle and for the F and G data value cycles. This can be clearly seen from the data values of the D0 waveform 46 shown in Fig. 4.

With respect to the multiplexer 33, when the Enable 1 signal 44 is high and the clock A signal goes high, the data value at the output of flip flop 31 is sampled by flip flop 37 and gated through to the output of flip flop 37, which corresponds to the D1 signal 47. When the Enable 1 signal 44 is low, the output of the flip flop 37 is maintained at its current value and thus the D1 signal 47 is maintained at that same value. The Enable 1 signal 44 goes low before the rising edge of clock A that occurs one clock A cycle before when the rising edges of clock A and clock B coincide. It goes high before the rising edge of clock A that occurs two clock A cycles after the coinciding rising edges of clocks A and B. The low time is fixed at three clock A cycles, but the high time will vary with the ratio of the clock frequencies.

The flip flop 37 delays the Data In signal value output from flip flop 31 by one clock A cycle. The first value of the Data In signal 41 that appears in the D1 signal waveform 47 in Fig. 4 is the B value. The Enable 1 signal 44 is low for the next three clock A cycles after the C value is gated through to the output of the flip flop 37, the D1 signal 41 maintains the C data value for four successive clock A cycles (i.e., for the C

data value cycle, for the D data value cycle and for the E data value cycle). This is indicated in the D0 waveform 47 shown in Fig. 4.

The Select signal 48 is timed to control the multiplexer 38 to select either the D0 or D1 signal to be output to the clock A frequency domain as Data Out 49. The Select signal 48 goes low just after the coinciding rising edges of clocks A and B. It then goes high just after the rising edge of clock A that is two cycles after when the rising edges of clock A and clock B coincided. The low time of the Select signal 48 is fixed at two clock A cycles, but the high time will vary based on the ratio of the clock frequencies. It is easy to see from the waveforms D0 46, D1 47 and Select 48 in Fig. 4 how the Data Out waveform 49 contains the data values shown. It can be seen from the Data Out waveform 49 that the C data value is output for two consecutive clock A cycles. As stated above, this is not a problem because the downstream logic (i.e., the logic of the clock A domain) knows when this will occur because it knows when the rising edges of the clocks will coincide and thus does not try to interpret the C data twice.

One of the advantages of the synchronizer of the present invention is that, because it uses flip flops instead of latches, the upstream logic is provided one full clock cycle for a signal to propagate through the upstream logic in the Data In path. As stated above, in previous synchronizer designs, latches have been used rather than flip flops, which resulted in variations in the amount of time available to the upstream logic. These variations, in turn, resulted in reduced setup time margins for the upstream logic.

Flip flops only change their outputs upon the occurrence of a particular triggering event, such as the rising edge of a clock. For example, a typical flip flop has a master-slave design and will sample the input while the clock is low, but the output will not

change to reflect the sampled input until the rising edge of the clock. In contrast, whatever is on the input of a latch is reflected on the output of the latch when the latch is in the "transparent mode". In the non-transparent mode of the latch, the output holds whatever value was on the output at the time that the signal that triggered the latch was removed. The transparent nature of the latch can potentially cause problems in the upstream logic in terms of the amount of time that the upstream logic has to make the Data In value valid. In other words, it is possible that the upstream logic will not be able to meet setup time requirements.

Because flip flops are used in the synchronizer design of the present invention, the Data In value does not have to be valid until the very end of a clock cycle, since it will not be sampled until the rising edge of the next clock cycle. Therefore, the setup time requirements only need to take into account the rising edge of the clock cycle of the frequency domain of the upstream logic. Another advantage of this large setup time margin is that it allows more logic to be placed in the path in the upstream logic that is feeding Data In of the input flip flop of the synchronizer.

Another advantage of the synchronizer of the present invention is that, because flip flops are used in the synchronizer, more skew is allowed between the two clocks. This is due to the fact that data is only being allowed to change on the edge of a clock, as opposed to being allowed to change during the time period when the clock is high and/or during the time period when the clock is low. For example, in the embodiment of Fig. 1, D1 is only changing on the rising edge of clock A and D0 is only changing on the falling edge of clock A. This allows for a larger clock skew margin than with previous synchronizer designs.

Yet another advantage of the synchronizer of the present invention is that Data Out is provided a full clock cycle before it needs to be valid. This, in turn, provides the downstream logic in the path of Data Out with a full clock cycle before it needs to be ready to receive data, which results in greater setup time margins for the downstream logic. As with the upstream logic, this allows more logic to be placed in the path of Data Out, if desired.

It should be noted that the present invention has been described with reference to particular embodiments. However, as will be understood by persons skilled in the art in view of the discussion provided herein, the present invention is not limited to the particular embodiments described herein. For example, it is well known that logic can be implemented in different ways and with different components to achieve the same result. Therefore, it will be understood that the particular logic designs shown in Figs. 1 and 3 have logical equivalents that are also covered by the present invention although they have not been specifically described herein. For example, flip flops can be added to either of the circuits shown in Figs. 1 and 3 without changing the logical results, albeit the latency of the circuits might be changed. Also, inverters can be used in various ways without changing logical end results. These and other modifications are within the scope of the present invention.